2

Two Processor Conservative Simulation Analysis

Robert E. Felderman and Leonard Kleinrock

ISI/RS-92-299
October 1992

University of Southern California
Information Sciences Institute
4676 Admiralty Way, Marina del Rey, CA 90292-6695
310-822-1511

This research was supported by the Defense Advanced Research Projects Agency under contract number MDA 903-87-C0663, Parallel Systems Laboratory and Ft. Huachuca contract DABT63-91-C-0001. This material is reprinted from the Proceedings of the 1992 SCS Western Simulation MultiConference on Parallel and Distributed Simulation 20-22 January 1992 The views and conclusions contained in this document are those of the authors and should not be interpreted as representing the official policies, either expressed or implied, of the Defense Advanced Research Projects Agency or the U.S. Government.

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

1. AGENCY USE ONLY *(Leave blank)*

2. REPORT DATE
1992, October

3. REPORT TYPE AND DATES COVERED
Research Report

4. TITLE AND SUBTITLE
Two Processor Conservative Simulation Analysis

5. FUNDING NUMBERS
C-MDA 903-87-C0663
C-DABT 63-91-C0001

6. AUTHOR(S)
Felderman, R.E.; Kleinrock, L.

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)
USC/Information Sciences Institute
4676 Admiralty Way
Marina del Rey, CA 90292-6695

8. PERFORMING ORGANIZATION REPORT NUMBER
ISI/RS-92-299

9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)
DARPA
3701 N. Fairfax Drive
Arlington, VA 22203-1714

10. SPONSORING/MONITORING AGENCY REPORT NUMBER

11. SUPPLEMENTARY NOTES

12a. DISTRIBUTION/AVAILABILITY STATEMENT
Approved for public release; distribution is unlimited

12b. DISTRIBUTION CODE

13. ABSTRACT *(Maximum 200 words)*

We present some new models and their exact analysis for the problem of two processors running a conservative distributed simulation protocol. The models show that lookahead is very useful in gaining performance, but only if the processors are well balanced in processing capacity. The models allow quantitative evaluation of the improvement in speedup attributed to null messages, as well as the degradation due to a cost for breaking deadlocks. Finally, a conservative system with "free" null messages and a small amount of lookahead is shown to outperform a Time Warp system with no cost for state saving or rollback.

14. SUBJECT TERMS
Distributed Simulation, Conservative, Optimistic, Time Warp, Performance Analysis

15. NUMBER OF PAGES
13

16. PRICE CODE

17. SECURITY CLASSIFICATION OF REPORT	18. SECURITY CLASSIFICATION OF THIS PAGE	19. SECURITY CLASSIFICATION OF ABSTRACT	20. LIMITATION OF ABSTRACT
Unclassified	Unclassified	Unclassified	UL

GENERAL INSTRUCTIONS FOR COMPLETING SF 298

The Report Documentation Page (RDP) is used in announcing and cataloging reports. It is important that this information be consistent with the rest of the report, particularly the cover and title page. Instructions for filling in each block of the form follow. It is important to ***stay within the lines*** to meet ***optical scanning requirements***.

Block 1. Agency Use Only *(Leave blank)*.

Block 2. Report Date. Full publication date including day, month, and year, if available (e.g. 1 Jan 88). Must cite at least the year.

Block 3. Type of Report and Dates Covered. State whether report is interim, final, etc. If applicable, enter inclusive report dates (e.g. 10 Jun 87 - 30 Jun 88).

Block 4. Title and Subtitle. A title is taken from the part of the report that provides the most meaningful and complete information. When a report is prepared in more than one volume, repeat the primary title, add volume number, and include subtitle for the specific volume. On classified documents enter the title classification in parentheses.

Block 5. Funding Numbers. To include contract and grant numbers; may include program element number(s), project number(s), task number(s), and work unit number(s). Use the following labels:

C -	Contract	PR -	Project
G -	Grant	TA -	Task
PE -	Program Element	WU -	Work Unit Accession No.

Block 6. Author(s). Name(s) of person(s) responsible for writing the report, performing the research, or credited with the content of the report. If editor or compiler, this should follow the name(s).

Block 7. Performing Organization Name(s) and Address(es). Self-explanatory.

Block 8. Performing Organization Report Number. Enter the unique alphanumeric report number(s) assigned by the organization performing the report.

Block 9. Sponsoring/Monitoring Agency Name(s) and Address(es). Self-explanatory.

Block 10. Sponsoring/Monitoring Agency Report Number. *(If known)*

Block 11. Supplementary Notes. Enter information not included elsewhere such as: Prepared in cooperation with...; Trans. of...; To be published in.... When a report is revised, include a statement whether the new report supersedes or supplements the older report.

Block 12a. Distribution/Availability Statement. Denotes public availability or limitations. Cite any availability to the public. Enter additional limitations or special markings in all capitals (e.g. NOFORN, REL, ITAR).

DOD -	See DoDD 5230.24, "Distribution Statements on Technical Documents."
DOE -	See authorities.
NASA -	See Handbook NHB 2200.2.
NTIS -	Leave blank.

Block 12b. Distribution Code.

DOD -	Leave blank.
DOE -	Enter DOE distribution categories from the Standard Distribution for Unclassified Scientific and Technical Reports.
NASA -	Leave blank.
NTIS -	Leave blank.

Block 13. Abstract. Include a brief *(Maximum 200 words)* factual summary of the most significant information contained in the report.

Block 14. Subject Terms. Keywords or phrases identifying major subjects in the report.

Block 15. Number of Pages. Enter the total number of pages.

Block 16. Price Code. Enter appropriate price code *(NTIS only)*.

Blocks 17. - 19. Security Classifications. Self-explanatory. Enter U.S. Security Classification in accordance with U.S. Security Regulations (i.e., UNCLASSIFIED). If form contains classified information, stamp classification on the top and bottom of the page.

Block 20. Limitation of Abstract. This block must be completed to assign a limitation to the abstract. Enter either UL (unlimited) or SAR (same as report). An entry in this block is necessary if the abstract is to be limited. If blank, the abstract is assumed to be unlimited.

Two Processor Conservative Simulation Analysis*

Robert E. Felderman†
USC/Information Sciences Institute
4676 Admiralty Way
Marina del Rey, CA 90292-6695
feldy@isi.edu

Leonard Kleinrock
UCLA Computer Science Department
3732L Boelter Hall
Los Angeles, CA 90024-1596
lk@cs.ucla.edu

Abstract

We present some new models and their exact analysis for the problem of two processors running a conservative distributed simulation protocol. The models show that lookahead is very useful in gaining performance, but only if the processors are well balanced in processing capacity. The models allow quantitative evaluation of the improvement in speedup attributed to null messages, as well as the degradation due to a cost for breaking deadlocks. Finally, a conservative system with "free" null messages and a small amount of lookahead is shown to outperform a Time Warp system with no cost for state saving or rollback.

1 Introduction

Conservative methods of Discrete Event Simulation are based on the work of Chandy, Misra, Bryant and others [1; 2; 3], and and excellent overview of the techniques involved may be found in [4]. Where TW proceeds ahead as fast as it can, only rolling back when a mistake is found, conservative methods allow an LP to proceed forward only when it is sure that it is performing correct computation. That is, conservative methods use blocking for synchronization, while optimistic techniques use state saving and rollback.

Kleinrock [5] and Felderman [6] have examined the performance of an optimistic method of distributed simulation, Time Warp (TW), running on two processors. In this paper we create models for a two processor system using a conservative synchronization algorithm rather than Time Warp. The emphasis is to create a model for a conservative algorithm that we may directly compare to the previously published models for Time Warp.

2 Previous Analytical Work

There has been a great deal of work in the area of conservative simulation. The bulk of it has been in creating and empirically evaluating the performance of a particular "flavor" of conservative synchronization. Our interest is in analytical results. Wagner and Lazowska [7; 8] provide techniques for bounding the speedup for simulation of queueing network models and discuss optimizations of conservative techniques. Lin, Lazowska and Baer [9] examine conservative simulation of systems without lookahead. They develop a new algorithm and provide an analytical model to estimate performance. In [10] Lin and Lazowska compare the performance of Time Warp and Chandy-Misra (conservative) simulation and derive sufficient conditions for Time Warp to outperform conservative simulation. Nicol [11; 12] provides performance bounds on a new conservative algorithm. Lubachevsky [13] shows that the "bounded lag" algorithm scales efficiently as the problem size grows.

Our effort in this paper is to provide a simple model for a conservative algorithm running on two processors so as to better understand the maximum improvement that can be gained by using null messages and exploiting lookahead. We also address the cost of deadlock detection and recovery to evaluate its impact on the performance of the algorithm. Finally, we compare the conservative approach to a previously published model for Time Warp [5] and show when the conservative approach outperforms Time Warp and vice versa.

*This work was supported by the Defense Advanced Research Projects Agency under Contract MDA 903-87-C0663, Parallel Systems Laboratory. To appear in the Proceedings of the 6th Workshop on Parallel and Distributed Simulation, January, 1992.

†The first author was also supported by DARPA contract number DABT63-91-0001. The views and conclusions contained in this document are those of the author(s) and should not be interpreted as representing the official policies, either expressed or implied, of the Defense Advanced Research Projects Agency or the U.S. Government.

3 The Model

We now describe our model for the conservative method of synchronization. Our goal is to create a model that can easily be compared to the previous models created for Time Warp [5]. Assume we have two processes each executing on its own processor (P_1, P_2). We use a continuous time, discrete state model, assuming that each process/processor advances along its own virtual time (simulated time) axis visiting only the integers. Each process takes an exponential amount of time to execute an event and advances exactly one step forward in virtual time (along its axis) after finishing the event. After advancing, each processor will send a synchronization message to the other processor with a given probability $q_i (i = 1, 2)$. Since the synchronization is conservative, no process can perform work at virtual time v until it is sure that the other processor will not send it a message time stamped with a virtual time less than or equal to v.

As was done in [5] we exploit the Markov process defined as the difference in virtual time (position on the axes) of the two processes, and find the probability that one processor is ahead of the other by a distance k. Note that $| k | \leq 1$ for unimproved conservative systems.

Here are the parameters of the model, chosen to correspond with those in [5].

$$\begin{aligned}
\lambda_i &= \text{rate at which processor } i \text{ executes events} \\
a &= \frac{\lambda_1}{\lambda_1 + \lambda_2} \\
\overline{a} &= \frac{\lambda_2}{\lambda_1 + \lambda_2} = 1 - a \\
q_i &= \text{P[} i^{th} \text{ proc. sends a msg. after advancing]} \\
\overline{q}_i &= 1 - q_i
\end{aligned}$$

We now examine several models for two processor conservative simulation.

4 A System Without Null Messages

We first solve a model where the processors do not send null messages. We assume that when a deadlock occurs it is detected and corrected after an exponential delay with mean $d/(\lambda_1 + \lambda_2)$. If $d = 0$ then a deadlock is broken instantaneously, while $d \rightarrow \infty$ means that deadlock detection and correction takes an infinite amount of time. This system can be described by a Markov chain with the following state description.

$$(D, t_1, t_2)$$

$$\begin{aligned}
D &= \text{Actual virtual time diff. between } P_1 \text{ and } P_2 \\
t_1 &= P_1\text{'s belief about the virtual time difference} \\
t_2 &= P_2\text{'s belief about the virtual time difference}
\end{aligned}$$

Discrepancies arise between D, t_1 and t_2 when the processors don't inform each other about state changes. This happens often when the processors are unlikely to send messages (small q_i). When a processor thinks it is ahead, it does not try to advance further. When both processors believe they are leading, we have a deadlock. The state diagram for this system is shown in Figure 1. Each state is labeled with its state description (D, t_1, t_2) and an alphanumeric label for calculation of the steady-state probabilities.

If the processors start out at the same virtual time v (state 0), eventually, one (say P_1) advances to $v+1$ and may send a message to P_2 (state D). Since a conservative synchronization mechanism is being employed, P_1 must wait to see if P_2 will send it a message with virtual time $v+1$. Its only choice is to wait until the lagging processor (P_2) advances, at which point that processor will "flip a coin" to decide whether to send a message. If a message is sent, P_1 receives it and is able to continue processing again (state 0). If a message isn't sent, P_1 thinks it is still ahead of the other processor and will not continue processing (state F). If P_2 were to advance again and not send a message, it would think (correctly) it was now ahead of P_1 and stop processing (state G). At this point we have a deadlock that must be broken. Deadlock detection and recovery algorithms have been discussed in [4]. Essentially, we break the deadlock by letting each processor know where it is relative to the other processor. In this example, P_1 would learn it was behind and begin processing, thus breaking the deadlock. If, on the other hand, each processor is able to notify the other that it has advanced its local clock, then the lagging processor is able to advance whether or not a "data" message is sent. This latter type of notification is referred to as the "null message" technique that is used to speed up conservative models. When used, we assume that this information (null messages) is propagated without cost.

The balance equations for this system can be derived from Figure 1.

$$\begin{aligned}
\lambda_1 p_A &= \lambda_2 \overline{q}_2 p_0 \\
\lambda_2 p_B &= \lambda_1 \overline{q}_1 p_0 \\
\lambda_1 p_C &= \lambda_2 q_2 p_0 + \lambda_2 q_2 p_F + \frac{\lambda_1 + \lambda_2}{d} p_G
\end{aligned}$$

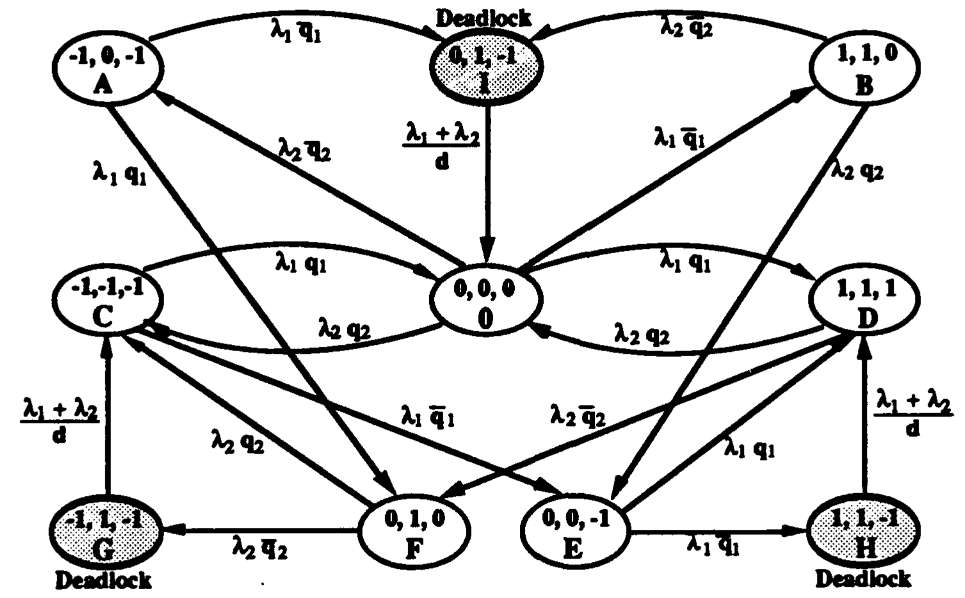

Figure 1: State diagram for conservative synchronization with no null messages and a cost for breaking deadlocks.

$$
\begin{aligned}
\lambda_2 p_D &= \lambda_1 q_1 p_0 + \lambda_1 q_1 p_E + \frac{\lambda_1+\lambda_2}{d} p_H \\
\lambda_1 p_E &= \lambda_2 q_2 p_B + \lambda_1 \bar{q}_1 p_C \\
\lambda_2 p_F &= \lambda_1 q_1 p_A + \lambda_2 \bar{q}_2 p_D \\
\frac{\lambda_1+\lambda_2}{d} p_G &= \lambda_2 \bar{q}_2 p_F \\
\frac{\lambda_1+\lambda_2}{d} p_H &= \lambda_1 \bar{q}_1 p_E \\
\frac{\lambda_1+\lambda_2}{d} p_I &= \lambda_1 \bar{q}_1 p_A + \lambda_2 \bar{q}_2 p_B \\
1 &= p_0 + p_A + p_B + p_C + p_D + \\
 & \quad p_E + p_F + p_G + p_H + p_I
\end{aligned}
$$

We first solve explicitly for $\{p_0, p_A, p_B, p_C, p_D, p_E, p_F, p_G, p_H, p_I\}$.

$$
p_A = \frac{\bar{a} p_0 \bar{q}_2}{a} \qquad p_B = \frac{a p_0 \bar{q}_1}{\bar{a}}
$$
$$
p_C = \frac{p_0 (1 - a q_2)}{a} \qquad p_D = \frac{p_0 (1 - \bar{a} q_1)}{\bar{a}}
$$
$$
p_E = \frac{p_0 \bar{q}_1}{a} \qquad p_F = \frac{p_0 \bar{q}_2}{\bar{a}}
$$
$$
p_G = d p_0 \bar{q}_2{}^2 \qquad p_H = d p_0 \bar{q}_1{}^2
$$
$$
p_I = d p_0 \bar{q}_1 \bar{q}_2
$$

$$
p_0 = (a\bar{a}) \Big/ \Big(1 - a\bar{a} + \bar{q}_1 + \bar{q}_2 + a\bar{a}d\left(3(1 - q_1 - q_2) - q_1 q_2 + (q_1 + q_2)^2\right)\Big)
$$

We then find the rate at which the two processors move forward in virtual time as

$$
R_2 = (\lambda_1+\lambda_2)p_0 + \lambda_1(p_A+p_C+p_E) + \lambda_2(p_B+p_D+p_F)
$$

We compare this rate to the equivalent single processor rate

$$
R_1 = \frac{\lambda_1+\lambda_2}{2}
$$

which is the average rate of virtual time progress if both processes are run on a single processor where additional synchronization is not necessary. One may think of the processes as being interleaved on a single processor.

Speedup is defined as the ratio of the two rates.

$$
\begin{aligned}
S &= \frac{R_2}{R_1} \\
 &= 2\left(p_0 + a(p_A + p_C + p_E) + \bar{a}(p_B + p_D + p_F)\right) \\
 &= 4 p_0 (3 - q_1 - q_2)
\end{aligned}
$$

For the simple case where $q_1 = q_2 = q$ the formula for speedup reduces to

$$
S = \frac{4a\bar{a}(3 - 2q)}{3 - 2q - a\bar{a}\left(1 - 3d\bar{q}^2\right)} \tag{1}
$$

and if the cost of breaking a deadlock is zero ($d = 0$) then the formula reduces to

$$S = \frac{4a\bar{a}\,(3-2q)}{3-a\bar{a}-2q} \tag{2}$$

and if $a = 1/2$ ($\lambda_1 = \lambda_2$), then

$$S = \frac{3-2q}{\frac{11}{4}-2q} \tag{3}$$

We show Equation 1 plotted versus a and q for various values of d in Figure 2. We note here that the conservative system with no cost for sending messages performs better as q, the interaction parameter, increases. This is in contrast to the Time Warp system [5] where speedup decreased as q increased. In the conservative system we are better off sending a large number of messages because the messages keep each process informed as to the virtual time progress of the other thus allowing potential parallelism to be exploited. When more messages are sent, the processors are less likely to be waiting due to lack of information and less likely to become deadlocked.

It is also clear from the figures that the cost of breaking a deadlock has a large impact on the performance if the probability of interaction (q_i) is small. This is to be expected, since the probability of deadlock is higher when the processes exchange information infrequently. We can take the derivative of speedup with respect to d (the cost of breaking deadlock) to quantify the effect of d on performance.

$$\frac{\partial S}{\partial d} = \frac{-12a^2\bar{a}^2\,(3-2q)\,\bar{q}^2}{\left(3-2q-a\bar{a}\,(1-3d\bar{q}^2)\right)^2}$$

We plot this function versus d and q for $a = 1/2$ in Figure 3 and see that changes in d have a large effect on S when q is small.

Returning again to speedup, we note that Equation 1 is only valid if $q_1 > 0$ and $q_2 > 0$. If both of these values are equal to zero (i.e., we never send messages), then speedup reduces to

$$S = \frac{4a\bar{a}}{\bar{a}(1+ad)+a^2} \tag{4}$$

and if $d = 0$ in this case we get

$$S = \frac{4a\bar{a}}{\bar{a}+a^2} \tag{5}$$

Coincidentally, this is also the formula we get if $q_1 = q_2 = 1$ or if $q_1, q_2 < 1$ and we always send null messages. For the $q_1 = q_2 = d = 0$ case, the system travels between states (A,0,B). In the null message case, the system travels between the states (C,0,D). Both systems produce the same probabilities and speedup. These systems produce the optimum speedup that can be gained from the conservative model. Equation 4 is plotted in Figure 4. and Equation 5 is plotted in Figure 5.

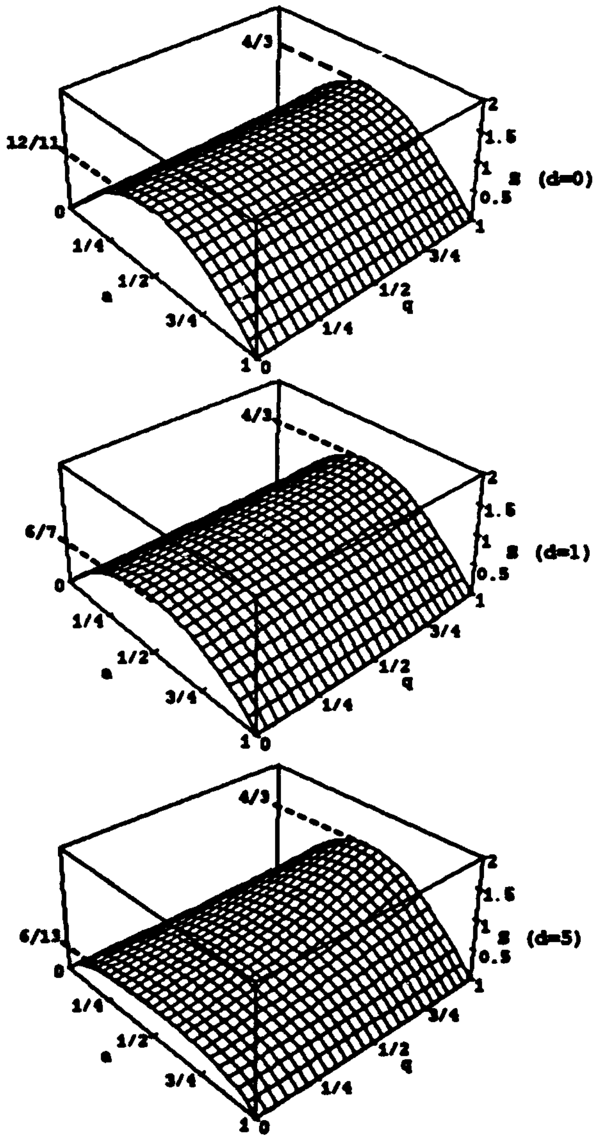

Figure 2: Speedup versus a and q for various values of d.

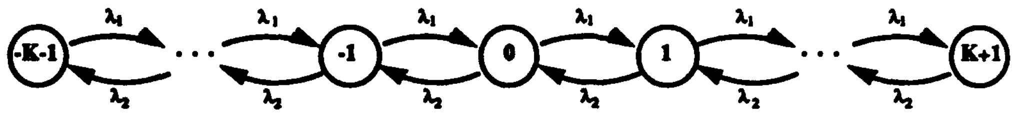

Figure 6: State diagram for a system with K-step lookahead.

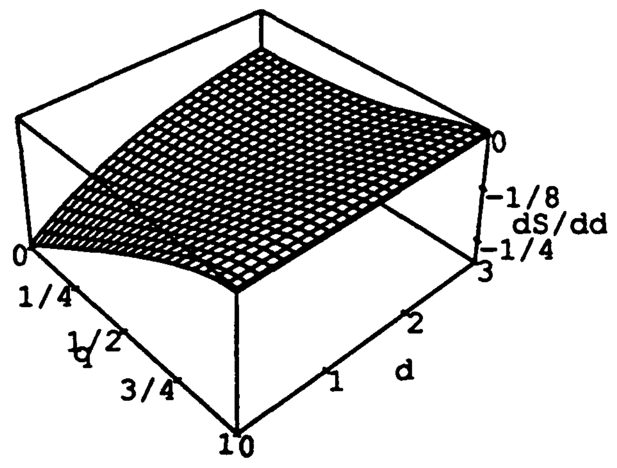

Figure 3: Derivative of speedup with respect to d (the cost of breaking a deadlock) versus q and d for $a = 1/2$.

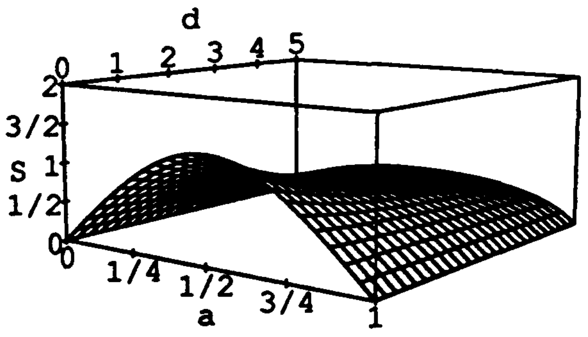

Figure 4: Speedup versus a and d for $q_1 = q_2 = 0$.

5 Lookahead

It has been noted by several researchers that exploiting lookahead is necessary to make conservative simulation a viable alternative to the optimistic approach [14; 15]. Lookahead is the ability of a logical process to predict its future behavior and especially its future output. In conservative simulation, when a process gives any downstream neighbor processes information about the arrival (or lack thereof) of future messages, the downstream processes are able to continue processing, thus enabling more parallelism in the system. The typical example of lookahead occurs in a FIFO queueing process. If jobs have a deterministic service time of S seconds (of simulated time), then if a server is empty at real time t and virtual time v, it can notify any downstream neigh-

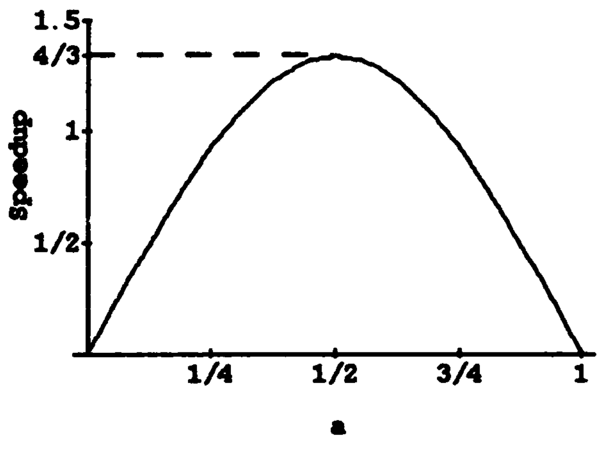

Figure 5: Maximum conservative speedup (i.e. for the system with null messages).

bor at real time t that no customer will arrive to this downstream queue with a virtual time stamp less than $v + S$. Therefore, recipient processes are able to execute any events they may have scheduled for virtual time less than $v + S$ (assuming no other input links).

5.1 Types of Lookahead

In order to formulate a model for a system using lookahead, we need to be very precise about what sort of future prediction is available. One example of this future prediction is that a process might always be able to inform the other processes of the virtual time of the next message it is going to send, but not the contents. With this sort of information, the receivers in a conservative system would be able to process all messages that had virtual times less than the time of the "scheduled" virtual time of the next message. In a two processor system each processor would execute messages with timestamps less than the virtual time of the "future" message, then wait for the arrival of that message. This system has the same performance as a TW system with no cost for state saving and rollback. TW is really forced to "wait" for the arrival of the message, but it is actually just performing useless work instead of waiting. Both systems return to processing useful work at the instant that the "straggler" message arrives.

Another type of lookahead is information that bounds the virtual time of future messages. The typical example (a FIFO queue) was given in the previous section. If we know something about the process that is being simulated, we may be able to provide information to downstream processes.

The type of lookahead that we use in our model was introduced by Nicol [11]. We can think of lookahead as the ability to transmit messages in our future to other processors. The farther into the future we are allowed to "precompute", the more lookahead we have. Nicol points out that there are two pieces of information contained in a lookahead message. The first is the virtual time of the pending message, the other is the actual contents of the message. Our previous example conveys only virtual time information while, in general, we could transmit both virtual time and data information. Nicol calls the lookahead with time and data information "full lookahead" while the time only message is "time lookahead". We use the idea of full lookahead in the next model due to its analytical tractability.

6 The Lookahead Model

Our definition of lookahead is based on our previous model that only allows processors to advance a single step in virtual time when advancing. By assuming that the processes have K-step full lookahead, each of the two processes is able to be at most $K+1$ units of virtual time (events) ahead of the other (as opposed to K *messages* ahead). Essentially we believe that a process is able to give the other process the content of any messages up to K virtual time units in the future. By assuming that null messages are used, each processor always knows its position relative to the other. Note that if $K = 0$, this model reverts to the simple no-lookahead model where a processor must wait when it gets ahead at all. The state diagram for this system is very simple and is shown in Figure 6. The balance equations for this system are:

$$\lambda_1 p_k = \lambda_2 p_{k+1} \qquad k = -K-1, \ldots, 0, \ldots, K$$

$$p_0 = 1 - \sum_{i=1}^{K+1} (p_i + p_{-i})$$

The solution is

$$p_k = \left(\frac{a}{\bar{a}}\right)^k p_0 \qquad k = -K-1, \ldots, 0, \ldots, K+1$$

$$p_0 = \frac{a^{K+1}(a-\bar{a})\bar{a}^{K+1}}{a^{3+2K} - \bar{a}^{3+2K}}$$

Speedup relative to the equivalent single processor implementation is

$$S = \frac{4a\bar{a}\left(a^{2+2K} - \bar{a}^{2+2K}\right)}{a^{3+2K} - \bar{a}^{3+2K}} \qquad a \neq \frac{1}{2} \qquad (6)$$

$$S = \frac{4(1+K)}{3+2K} \qquad a = \frac{1}{2} \qquad (7)$$

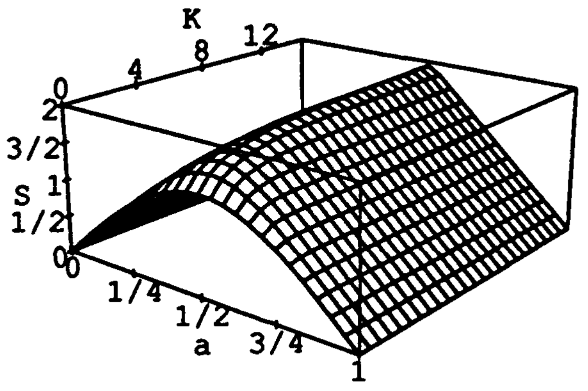

Figure 7: Speedup for a K-step lookahead conservative system.

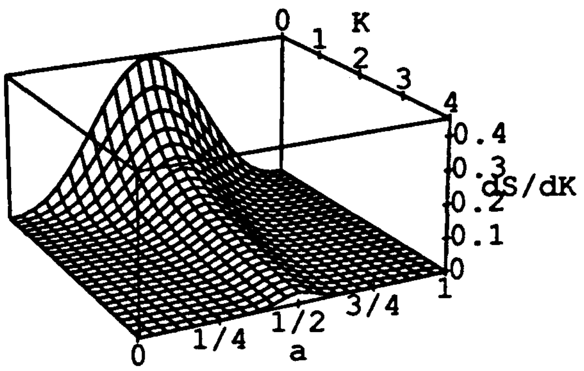

Figure 8: Derivative of speedup with respect to K.

Equation 6 is plotted versus a and K in Figure 7. We can see from this figure that lookahead is extremely useful when the processors are nearly balanced in processing speed ($a = 1/2$). In the imbalanced situation, the faster processor quickly runs out to its limit of K steps, then waits for the other processor to move forward before it can continue again. By taking the derivative of speedup with respect to K, we see this result more clearly. In Figure 8 we show $\partial S/\partial K$. When K is small and a is near 1/2, any change in K has a major effect on speedup, though once we move away from $a = 1/2$ or $K > 5$, the impact is significantly reduced. The moral of this story is to make sure the processes progress at nearly the same rate in virtual time or lookahead will be useless.

7 Comparison to Time Warp

We now make a direct comparison between the speedup results obtained from the previously published T'me Warp models and conservative models derived in the previous sections. To clearly display the tradeoffs, we compare simplified versions of each. Figure 9 shows the ratio of speedup for the conservative model using null messages but no lookahead to Time Warp with no cost for state saving and rollback [5]. It is clear that "free" Time Warp is always a winner since the ratio never exceeds one. The opti-

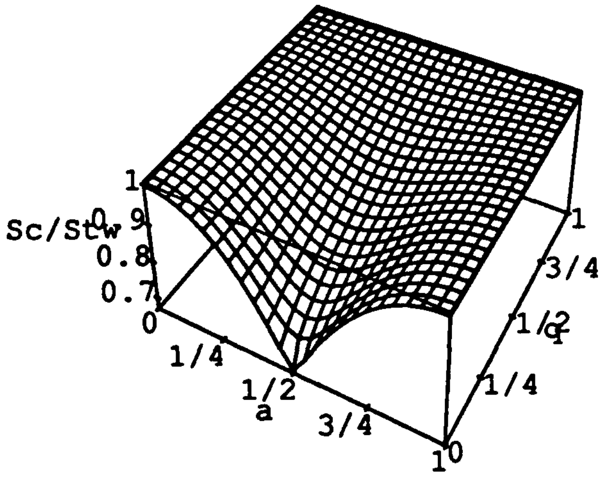

Figure 9: Ratio of conservative speedup (no lookahead) to "free" Time Warp speedup.

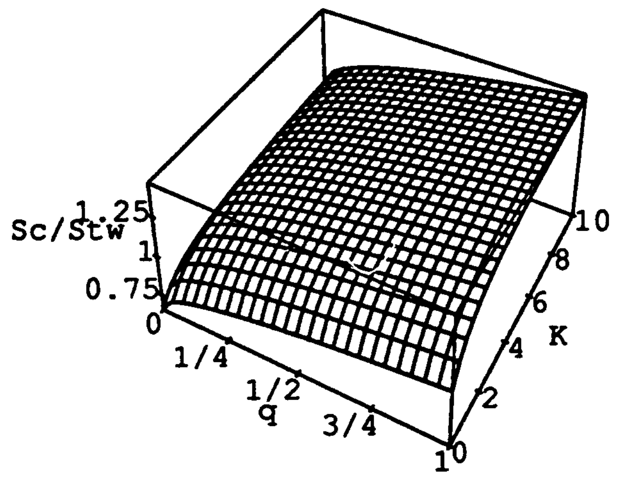

Figure 10: Ratio of conservative speedup with K-step lookahead to "free" Time Warp with no lookahead.

mistic approach with no cost for its aggressive computation is always better.

Let us now compare free TW to the conservative model with lookahead when both systems are operating at $a = 1/2$ and when the conservative system has K-step lookahead. Proponents of the optimistic approach point out that their systems work well regardless of whether lookahead is exploited. Our comparison is an attempt to see how well the conservative approach exploiting lookahead fares with respect to a Time Warp system that uses no lookahead. This ratio is plotted in Figure 10 and suggests that a little lookahead combined with null messages goes a long way. For almost any value of K greater than one, we see that the conservative model outperforms "free" Time Warp (ratio > 1). We find the threshold where the conservative approach beats TW by solving the following inequality for K.

$$\frac{S_{cons}}{S_{TW}} = \frac{(1+K)(2+\sqrt{q})}{3+2K} \geq 1 \qquad (8)$$

The condition for the conservative approach to beat Time Warp is

$$K \geq \frac{1-\sqrt{q}}{\sqrt{q}} \qquad (9)$$

For q (the interaction parameter) very small we need a large lookahead, but for $q > 0.1$, K only needs to be

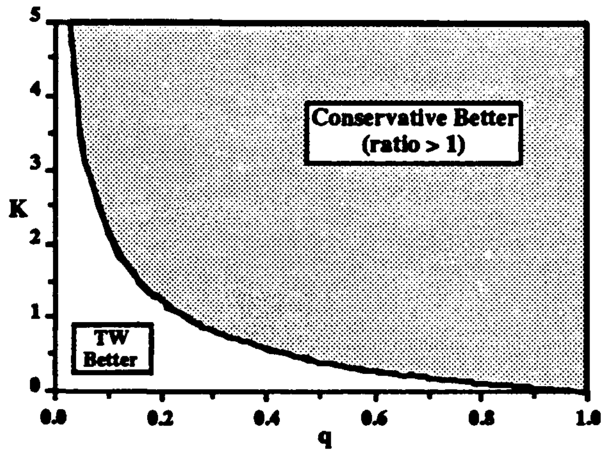

Figure 11: Area of the $q - K$ plane where the conservative approach with lookahead wins out.

1 or 2. Figure 11 shows the areas of the $q - K$ plane where the conservative approach beats "free" Time Warp. Note that if an optimistic system with no rollback and state saving costs is afforded the same lookahead as a conservative system with no cost for null message transmissions, the optimistic approach will always perform better since it is able to aggressively compute along the critical path for free.

8 Conclusions

This paper examined some simple two processor models for the conservative synchronization method. It showed that lookahead is very useful in gaining performance, but only if the processors are well balanced in processing capacity. The models allowed quantitative evaluation of the improvement attributed to null messages, as well as the degradation due to a cost for breaking deadlocks. Finally, a conservative system with "free" null messages and a small amount of lookahead was shown to outperform a Time Warp system with no cost for state saving or rollback. However, if they both incorporate lookahead, then TW is the winner. Unfortunately for the conservative approach, lookahead is not often easy to come by [14; 15]. A simple FIFO queueing system provides great lookahead, but add in preemptive-priority queueing and all the lookahead disappears. It may be unwise to utilize a synchronization mechanism that needs lookahead to perform well.

References

[1] K. Mani Chandy and Jayadev Misra. Dis-

tributed simulation: A case study in design and verification of distributed programs. *IEEE Transactions on Software Engineering*, SE-5(5), 1979.

[2] K.M. Chandy, Victor Holmes, and J. Misra. Distributed simulation of networks. *Computer Networks*, 3(2), 1979.

[3] R.E. Bryant. Simulation of packet communication architecture computer systems. Technical Report MIT,LCS,TR-188, Massachusetts Institute of Technology, Cambridge, Mass., 1977.

[4] Jayadev Misra. Distributed discrete-event simulation. *Computing Surveys*, 18(1):39–65, March 1986.

[5] Leonard Kleinrock. On distributed systems performance. In *Proceedings of the 7th ITC Specialist Seminar, Adelaide, Australia*. ITC, September 1989. (Also published in "Computer Networks and ISDN Systems" vol. 20, no.1-5, pp. 206-215, December 1990.).

[6] Robert E. Felderman and Leonard Kleinrock. Two processor time warp analysis: Some results on a unifying approach. In *Proceedings of the SCS Multiconference on Advances in Parallel and Distributed Simulation*, volume 23,1, pages 3–10. Society for Computer Simulation, January 1991.

[7] D.B. Wagner and E.D. Lazowska. Parallel simulation of queueing networks: Limitations and potentials. In *Proceedings of 1989 ACM SIGMETRICS and PERFORMANCE '89*, volume 17,1, pages 146–155, May 1989.

[8] David B. Wagner. Conservative parallel discrete-event simulation: Principles and practice. Technical Report 89-09-03, Department of Computer Science and Engineering, University of Washington, September 1989.

[9] Y-B. Lin, E.D. Lazowska, and J-L. Baer. Conservative parallel simulation for systems with no lookahead prediction. In *Proceedings of the SCS Multiconference on Distributed Simulation*, volume 22,1, pages 144–149. Society for Computer Simulation, January 1990.

[10] Yi-Bing Lin and Edward D. Lazowska. Optimality considerations for "time warp" parallel simulation. In *Proceedings of the SCS Multiconference on Distributed Simulation*, volume 22,1, pages 29–34. Society for Computer Simulation, January 1990.

[11] David M. Nicol. Parallel self-initiating discrete-event simulations. *Transactions on Modelling and Computer Simulation*, 1(1):24–50, January 1991.

[12] D. M. Nicol. The cost of conservative synchronization in parallel discrete event simulations. Technical Report 90-20, Institute for Computer Applications in Science and Engineering(ICASE), May 1990.

[13] Boris D. Lubachevsky. Scalability of the bounded lag distributed discrete event simulation. In *Proceedings of the SCS Multiconference on Distributed Simulation*, pages 100–107. The Society for Computer Simulation, July 1989.

[14] Richard M. Fujimoto. Lookahead in parallel discrete event simulation. In *International Conference on Parallel Processing*, 1988.

[15] D. M. Nicol. Parallel discrete-event simulation of fcfs stochastic queueing networks. *SIGPLAN Not.*, 23(9):124–137, September 1988.

Bob Felderman was born in Chicago, Illinois in 1962. He graduated Magna Cum Laude from Princeton University in 1984 with a double major in Electrical Engineering & Computer Science and Systems Engineering. After spending a year a Hughes Aircraft Co., he returned to the good life of academia, completed his Ph.D in Computer Science at UCLA in 1991 and was named one of three outstanding Ph.D. students by the School of Engineering and Applied Science. He is currently a Computer Scientist at USC/Information Sciences Institute where he is working on high speed local area networks, distributed systems and optimistic computing. He is also a lecturer in the UCLA Computer Science Department. In his spare time he can be found in the great outdoors usually with frisbee in hand. He only rarely refers to himself in the third person.